

# AN EFFICIENT ALGORITHM FOR UNWEIGHTED SPECTRAL GRAPH SPARSIFICATION\*

DAVID G. ANDERSON<sup>†</sup>, MING GU<sup>†</sup>, AND CHRISTOPHER MELGAARD<sup>†</sup>

**Abstract.** Spectral graph sparsification has emerged as a powerful tool in the analysis of large-scale networks by reducing the overall number of edges, while maintaining a comparable graph Laplacian matrix. In this paper, we present an efficient algorithm for the construction of a new type of spectral sparsifier, the *unweighted* spectral sparsifier. Given a general undirected and unweighted graph  $G = (V, E)$ , and an integer  $\ell < |E|$  (the number of edges in  $E$ ), we compute an unweighted graph  $H = (V, F)$  with  $F \subset E$  and  $|F| = \ell$  such that for every  $x \in \mathbb{R}^V$

$$\frac{x^T L_G x}{\kappa} \leq x^T L_H x \leq x^T L_G x,$$

where  $L_G$  and  $L_H$  are the Laplacian matrices for  $G$  and  $H$ , respectively, and  $\kappa \geq 1$  is a slowly-varying function of  $|V|$ ,  $|E|$  and  $\ell$ . This work addresses the open question of the existence of *unweighted* graph sparsifiers for unweighted graphs [3]. Additionally, our algorithm efficiently computes unweighted graph sparsifiers for weighted graphs, leading to sparsified graphs that retain the weights of the original graphs.

**Key words.** graph sparsification, spectral graph theory, spectral sparsification, unweighted graph sparsification

**AMS subject classifications.** 68R10, 90C35, 15A18, 15B34, 15B48

**1. Introduction.** Graph sparsification seeks to approximate a graph  $G$  with a graph  $H$  on the same vertices, but with fewer edges. Called a sparsifier,  $H$  requires less storage than  $G$  and serves as a proxy for  $G$  in computations where  $G$  is too large, evoking the effectiveness of sparsifiers in wide-ranging applications of graphs, including social networks, conductance, electrical networks, and similarity [7, 8, 15, 18]. In some applications, graph sparsification improves the quality of the graph, such as in the design of information networks and the hardwiring of processors and memory in parallel computers [4, 11]. Sparsifiers have also been utilized to find approximate solutions of symmetric, diagonally-dominant linear systems in nearly-linear time [4, 17, 18, 19].

Recent work on graph sparsification includes [1, 10, 16, 18, 20]. Batson, Spielman, and Srivastava [3] prove that for every graph there exists a spectral sparsifier where the number of edges is linear in the number of vertices. They further provide a polynomial-time, deterministic algorithm for the sparsification of weighted graphs, which could produce weights that differ greatly from the weights of the original graph. The work of Avron and Boutsidis [2] explores unweighted sparsification in the context of finding low-stretch spanning trees. They provide a greedy edge removal algorithm and a volume sampling algorithm with theoretical guarantees. In comparison, our novel greedy edge selection algorithm has tighter theoretical bounds for both spanning trees and in the more general context of unweighted graph sparsification.

Our work introduces a deterministic, greedy edge selection algorithm to calculate sparsifiers for weighted and unweighted graphs. Our algorithm selects a subset of edges for the sparse approximation  $H$ , without assigning or altering weights. While the Dual Set algorithms of [2, 3, 6] reweight all selected edges for computing *weighted*

\*This work was partially supported by NSF Award CCF-131912

<sup>†</sup>Department of Mathematics, University of California, Berkeley, California 94720 (anderson@math.berkeley.edu, mgu@math.berkeley.edu, melgaard@math.berkeley.edu).

*sparsifiers*, our algorithm produces *unweighted sparsifiers* for an unweighted input graph, and can create a weighted sparsifier for a weighted input graph by assigning the original edge weights to the sparsifier. Hence our concept of unweighted sparsification applies to both unweighted and weighted graphs. To formalize:

DEFINITION 1.1. Let  $G = (V, E, w)$  be a given graph<sup>1</sup>. We define an **unweighted sparsification** of  $G$  to be any graph of the form  $H = (V, F, w \odot \mathbb{I}_F)$ , where

$$\mathbb{I}_F(e) = \begin{cases} 1, & \text{if } e \in F \\ 0, & \text{otherwise} \end{cases}$$

is the indicator function and  $\odot$  is the Hadamard product, i.e.

$$(w \odot \mathbb{I}_F)(e) = \begin{cases} w_e, & \text{if } e = (u, v) \in F \\ 0, & \text{otherwise} \end{cases}.$$

Several definitions have been proposed for the notion in which a sparsifier approximates a dense graph. Benzcúr and Karger [5] introduced cut sparsification, where the sum of the weights of the edges of a cut dividing the set of vertices is approximately the same for the dense graph and the sparsifier. Spielman and Teng [20] proposed spectral sparsification, a generalization of cut sparsification, which seeks sparsifiers with a Laplacian matrix close to that of the input graph. We follow the work of [3, 20] and base our work on spectral sparsification, for which we now present a rigorous definition.

Given an undirected graph  $G = (V, E, w)$ , define the signed edge-vertex incidence matrix  $B_G \in \mathbb{R}^{E \times V}$  as

$$(B_G)_{ej} = \begin{cases} -1, & \text{if } e = (u, v) \in E \text{ and } j = u \in V \\ 1, & \text{if } e = (u, v) \in E \text{ and } j = v \in V, \\ 0 & \text{otherwise} \end{cases}$$

where all edges are randomly assigned a direction, and  $e = (u, v) \in E$  is an edge from  $u$  to  $v$ . Define the diagonal weight matrix  $W_G \in \mathbb{R}^{E \times E}$

$$(W_G)_{ef} = \begin{cases} w_e, & \text{if } e = f \in E, \\ 0 & \text{otherwise} \end{cases}.$$

The Laplacian of the graph is

$$L_G = B_G^T W_G B_G.$$

Note that

$$x^T L_G x = \sum_{(u,v) \in E} w_{(u,v)} (x_u - x_v)^2$$

for a vector  $x \in \mathbb{R}^V$ . To compare Laplacians of graphs  $X$  and  $Y$  defined on the same set of nodes we denote

$$L_X \preceq L_Y \quad \text{if and only if} \quad x^T L_X x \leq x^T L_Y x, \quad \text{for all } x.$$

---

<sup>1</sup>Note that any unweighted graph  $G = (V, E)$  induces a weighted graph  $G = (V, E, w)$  where  $w_e = 1$  if  $e = (u, v) \in E$  and  $w_e = 0$  otherwise.

DEFINITION 1.2. *The graph  $H$  is a  $\kappa$ -approximation of  $G$  if*

$$\frac{1}{\kappa}L_G \preceq L_H \preceq L_G.$$

Because our unweighted sparsification algorithm does not change the weights of the edges kept in  $H$ , it is immediate that  $L_H \preceq L_G$ :

PROPOSITION 1.3. *If  $H$  is an unweighted sparsification of  $G$ , then*

$$L_H \preceq L_G.$$

*Proof.*

$$\begin{aligned} x^T L_H x &= \sum_{(u,v) \in F} w_{(u,v)} (x_u - x_v)^2 \\ &\leq \sum_{(u,v) \in F} w_{(u,v)} (x_u - x_v)^2 + \sum_{(u,v) \in E \setminus F} w_{(u,v)} (x_u - x_v)^2 \\ &= x^T L_G x \end{aligned}$$

for all  $x \in \mathbb{R}^V$ .  $\square$

Our algorithm does not operate directly on the Laplacian matrix. Rather, we consider the SVD of  $W_G^{1/2} B_G$ .

$$(1.1) \quad W_G^{1/2} B_G = U_G^T \Sigma_G V_G,$$

where  $\Sigma_G$  is a diagonal matrix containing all non-zero singular values of  $W_G^{1/2} B_G$ ; and where  $U_G \in \mathbb{R}^{n \times m}$  is a row orthonormal matrix, with  $n = |V| - r$ , and  $r$  being the number of connected components in  $G$ . For the unweighted graph,  $W_G$  is simply the identity matrix.  $U_G$  plays a similar role to that of the matrix  $V_{n \times m}$  in [3] and the matrix  $Y$  in [2]. Our algorithm utilizes the column-orthogonality of  $U_G^T$ , highlighting the reason for not working directly with the Laplacian matrix. We note, nevertheless, that this algorithm can be adapted to any orthogonal decomposition of  $W_G^{1/2} B_G$ .

We are now in a position to present our main results. In section 2 we present the unweighted column selection algorithm, as well as the spectral bounds for the sparsifiers it calculates. Section 3 provides supporting theory. Comparisons to other modern algorithms are made in section 4. In section 5 we demonstrate one application of graph sparsification, graph visualization, by applying our algorithm to real autonomous systems data. Some observations and concluding remarks are offered in sections 6 and 7.

**2. The Unweighted Column Selection (UCS) Algorithm.** Our algorithm selects edges for a sparsifier based on the columns  $u_i$  of  $U_G$ ,

$$U_G = \begin{pmatrix} u_1 & u_2 & \cdots & u_m \end{pmatrix} \in \mathbb{R}^{n \times m},$$

where  $m = |E|$  is the number of edges, and  $n = |V| - r$ , as above. Therefore, the edges of  $G$  that are included in the sparsifier are exactly the columns of  $U_G$  that our algorithm selects. Denote the number of edges kept as  $\ell \stackrel{\text{def}}{=} |F|$ . Let  $\Pi_t$  denote the set of selected edges after  $t$  iterations.

We propose the following greedy algorithm for column selection on  $U_G$ . Initially set  $A_0 = 0_{n \times n}$  and  $\Pi_0 = \emptyset$ , and choose a constant  $T > 0$ . At step  $t \geq 0$ :

- Solve for the unique  $\lambda < \lambda_{\min}(A_t)$  such that

$$(2.1) \quad \text{tr}(A_t - \lambda I)^{-1} = T.$$

- Solve for the unique  $\hat{\lambda} \in (\lambda, \lambda_{\min}(A))$  such that

$$(2.2) \quad (\hat{\lambda} - \lambda) \left( m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \lambda} \right) = \frac{\sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}}{\sum_{j=1}^n \frac{1}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}},$$

where  $\lambda_j$  is the  $j^{\text{th}}$  largest eigenvalue of the symmetric matrix  $A_t$ .

- Find an index  $i \notin \Pi_t$  such that

$$(2.3) \quad \text{tr}(A_t - \hat{\lambda}I + u_i u_i^T)^{-1} \leq \text{tr}(A_t - \lambda I)^{-1}.$$

- Update  $A_t$  and  $\Pi_t$ .

While equations (2.1) and (2.2) are relatively straightforward to justify and solve, equation (2.3) requires careful consideration, and is the focus of much of section 3. Note that equation (2.1) can be solved in  $O(n^3)$  operations, equation (2.2) in  $O(n)$  operations, and equation (2.3) in  $O(n^2 m)$  operations. This last complexity count follows because testing the inequality scales with  $O(n^2)$ , and potentially all remaining indices must be tested. Thus the total complexity of selecting  $\ell$  columns is  $O(\ell n^2 m)$ .

While this procedure will work for any  $T > 0$ , we will show that an effective choice is

$$T = \hat{T}^* \left( 1 + F(\hat{T}^*) \right),$$

where

$$F(\hat{T}) = \left[ \left( 1 - \frac{n}{\hat{T}} \right) \frac{\ell}{m - \frac{\ell-1}{2} + \hat{T} - n} - \frac{n}{\hat{T}} \right],$$

and where  $\hat{T}^*$  is the minimizer of  $F(\hat{T})$ , given as

$$\hat{T}^* = \frac{n(m + \frac{\ell+1}{2} - n) + \sqrt{n\ell(m - \frac{\ell-1}{2})(m + \frac{\ell+1}{2} - n)}}{\ell - n}.$$

Our spectral bounds are derived using this choice of  $T$ . We summarize this procedure in the Unweighted Column Selection algorithm.

Theorem 2.1 below confirms the correctness of the Unweighted Column Selection Algorithm. This theorem, along with other properties of the UCS algorithm, will be discussed and proved in Section 3.

**THEOREM 2.1.** *Let  $G = (V, E, w)$  and let  $n < \ell < m$ . Then the sparsified graph  $H$  produced by the UCS algorithm satisfies*

$$\frac{1}{\kappa} L_G \preceq L_H \preceq L_G,$$

where

$$(2.4) \quad \frac{1}{\kappa} = \frac{(\ell - n)^2}{\left( \sqrt{n(m + \frac{\ell+1}{2} - n)} + \sqrt{\ell(m - \frac{\ell+1}{2})} \right)^2 + (\ell - n)^2}.$$

ALGORITHM: UNWEIGHTED COLUMN SELECTION (UCS)

**Inputs:**  $G = (V, E, w)$ ,  $T > 0$ ,  $\ell$ .

**Outputs:**  $H_{uw} = (V, F, w \odot \mathbb{I}_F)$

- 1: Calculate the column-orthogonal matrix  $U_G^T$
- 2: Set  $A_0 = 0_{n \times n}$ ,  $\Pi_0 = \emptyset$
- 3: **for**  $t = 0, \dots, \ell - 1$  **do**
- 4:   Solve for  $\lambda$  using equation (2.1)
- 5:   Calculate  $\hat{\lambda}$  using equation (2.2)
- 6:   Find  $i \notin \Pi_t$  such that inequality (2.3) is satisfied
- 7:   Update  $A_{t+1} = A_t + u_i u_i^T$
- 8:   Update  $\Pi_{t+1} = \Pi_t \cup \{i\}$
- 9: **end for**
- 10: Let  $F = \Pi_\ell$  be the selected edges

**3. Correctness and Performance of the UCS Algorithm.** The goal of this section is to prove Theorem 2.1. Section 3.1 establishes that the UCS algorithm is well-defined. Section 3.2 proves a lower bound for the minimum singular value of the submatrix selected by the UCS algorithm, and provides a good choice for the input parameter  $T$ . In section 3.3, the UCS algorithm is shown to be a graph sparsification algorithm.

**3.1. The Existence of a Solution to Equation (2.3).** The next two lemmas show that equation (2.3) always has a solution.

LEMMA 3.1. *At a given iteration  $t$  in the UCS algorithm, at step 6 define*

$$f(x) \stackrel{\text{def}}{=} (x - \lambda) \left[ m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \lambda} \right] - \frac{\sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - x)}}{\sum_{j=1}^n \frac{1}{(\lambda_j - \lambda)(\lambda_j - x)}}.$$

*Then there exists  $\hat{\lambda}$ , with  $\lambda < \hat{\lambda} < \lambda_n$ , such that  $f(\hat{\lambda}) = 0$ . Furthermore,*

$$(3.1) \quad 0 < (\hat{\lambda} - \lambda) \left[ \frac{\sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})^2}}{\sum_{j=1}^n \frac{1}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}} - (\hat{\lambda} - \lambda) \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right].$$

*Proof.* Clearly  $f(\lambda) < 0$ . Although  $f$  is undefined at  $\lambda_n$ , let  $\lambda_n^\epsilon := \lambda_n - \epsilon$ , where  $\epsilon > 0$ . Note that

$$\lim_{\epsilon \rightarrow 0^+} \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \lambda_n^\epsilon)} \right) / \left( \sum_{j=1}^n \frac{1}{(\lambda_j - \lambda)(\lambda_j - \lambda_n^\epsilon)} \right) = 1 - \lambda_n$$

because the last term in each sum will dominate the rest of the sum. Furthermore,

$$\begin{aligned} \lim_{\epsilon \rightarrow 0^+} (\lambda_n^\epsilon - \lambda) \left[ m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \lambda} \right] &= 1 - \lambda_n + \lim_{\epsilon \rightarrow 0^+} (\lambda_n^\epsilon - \lambda) \left[ m - t + \sum_{j=1}^{n-1} \frac{1 - \lambda_j}{\lambda_j - \lambda} \right] \\ &> 1 - \lambda_n. \end{aligned}$$

Hence for small  $\epsilon > 0$ , we have  $f(\lambda_n^\epsilon) > 0$ , and, therefore,  $\hat{\lambda}$  exists, with  $\lambda < \hat{\lambda} < \lambda_n$ , and  $f(\hat{\lambda}) = 0$  via the Intermediate Value Theorem. Note that if there exists

$0 < \gamma < n$  such that  $\lambda_\gamma = \lambda_{\gamma+1} = \dots = \lambda_n$ , then we repeat the same argument replacing the expression  $1 - \lambda_n$  with  $\sum_{j=\gamma}^n 1 - \lambda_j = (n - \gamma + 1)(1 - \lambda_n)$ .

Now we prove inequality (3.1). We use the following version of the Cauchy-Schwartz formula: for  $a_j, b_j \geq 0$  then  $(\sum a_j b_j)^2 \leq (\sum a_j^2 b_j) (\sum b_j)$ . Consequently

$$\begin{aligned}
& \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right)^2 \\
& \leq \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})^2} \right) \left( 0 + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \\
& < \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})^2} \right) \left( \overbrace{m-t}^{>0} + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \\
& = \frac{1}{(\hat{\lambda} - \lambda)} \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})^2} \right) \left( \frac{\sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}}{\sum_{j=1}^n \frac{1}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})}} \right),
\end{aligned}$$

where the last step comes from  $f(\hat{\lambda}) = 0$ . The strict inequality above holds because  $m - t \geq m - \ell + 1 \geq 1$ . After some simple algebra,

$$(\hat{\lambda} - \lambda) \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} < \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})^2} \right) \bigg/ \left( \sum_{j=1}^n \frac{1}{(\lambda_j - \lambda)(\lambda_j - \hat{\lambda})} \right),$$

which implies our desired inequality because  $0 < \hat{\lambda} - \lambda$ .  $\square$

Next, we show that our algorithm is well defined in the sense we can always find a new index  $i \notin \Pi_t$  for each iteration that satisfies (6).

**LEMMA 3.2.** *An index  $i \notin \Pi_t$  can always be found to satisfy line (6) of the UCS algorithm for  $0 \leq t < \ell$ .*

*Proof.* Note the two following partial fraction results

$$(3.2) \quad \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} = \frac{1}{\lambda_j - \hat{\lambda}} - \frac{1}{\lambda_j - \lambda}$$

$$(3.3) \quad \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)^2} + \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} = \frac{1}{(\lambda_j - \hat{\lambda})^2}.$$

Using the fact that  $f(\hat{\lambda}) = 0$ , followed by the inequality of Lemma 3.1, we have

$$\begin{aligned}
& (\hat{\lambda} - \lambda) \left[ m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right] \\
&= \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) / \left( \sum_{j=1}^n \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) + 0 \\
&< \left( \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) / \left( \sum_{j=1}^n \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) \\
&\quad + (\hat{\lambda} - \lambda) \left[ \frac{\sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)^2}}{\sum_{j=1}^n \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)}} - (\hat{\lambda} - \lambda) \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right] \\
&= \left( (\hat{\lambda} - \lambda) \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)^2} + \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) \\
&\quad / \left( \sum_{j=1}^n \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) - (\hat{\lambda} - \lambda)^2 \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \\
&= \frac{\sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})^2}}{\sum_{j=1}^n \frac{1}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)}} - (\hat{\lambda} - \lambda) \left( \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} - \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \lambda} \right),
\end{aligned}$$

where the last line follows from equations (3.2) and (3.3). After some rearranging:

$$\left( m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \left( \sum_{j=1}^n \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) < \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})^2}.$$

This inequality can be rewritten using the trace property  $\mathbf{tr}(xy^T) = y^T x$  and the identity  $\sum_{i \notin \Pi_t} u_i u_i^T = \sum_{i=1}^m u_i u_i^T - \sum_{i \in \Pi_t} u_i u_i^T = I_n - A_t$ :

$$\begin{aligned}
& \left( \sum_{i \notin \Pi_t} 1 + u_i^T (A_t - \hat{\lambda} I)^{-1} u_i \right) \left( \mathbf{tr} (A_t - \hat{\lambda} I)^{-1} - \mathbf{tr} (A_t - \lambda I)^{-1} \right) \\
&= \left( m - t + \sum_{i \notin \Pi_t} \mathbf{tr} \left[ (A_t - \hat{\lambda} I)^{-1} u_i u_i^T \right] \right) \left( \sum_{j=1}^n \frac{1}{\lambda_j - \hat{\lambda}} - \sum_{j=1}^n \frac{1}{\lambda_j - \lambda} \right) \\
&= \left( m - t + \mathbf{tr} \left[ (A_t - \hat{\lambda} I)^{-1} (I - A_t) \right] \right) \sum_{j=1}^n \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \\
&= \left( m - t + \sum_{j=1}^n \frac{1 - \lambda_j}{\lambda_j - \hat{\lambda}} \right) \left( \sum_{j=1}^n \frac{\hat{\lambda} - \lambda}{(\lambda_j - \hat{\lambda})(\lambda_j - \lambda)} \right) \\
&< \sum_{j=1}^n \frac{1 - \lambda_j}{(\lambda_j - \hat{\lambda})^2} \\
&= \mathbf{tr} \left( (A_t - \hat{\lambda} I)^{-2} (I - A_t) \right) \\
&= \sum_{i \notin \Pi_t} u_i^T (A_t - \hat{\lambda} I)^{-2} u_i.
\end{aligned}$$

Moving terms to the right and dividing by  $\left(\mathbf{tr}(A_t - \hat{\lambda}I)^{-1} - \mathbf{tr}(A_t - \lambda I)^{-1}\right) > 0$  (because  $\hat{\lambda} > \lambda$ ) gives

$$\sum_{i \notin \Pi_t} \left( \frac{u_i^T (A_t - \hat{\lambda}I)^{-2} u_i}{\mathbf{tr}(A_t - \hat{\lambda}I)^{-1} - \mathbf{tr}(A_t - \lambda I)^{-1}} - \left(1 + u_i^T (A_t - \hat{\lambda}I)^{-1} u_i\right) \right) > 0.$$

For this to be true, there must exist an  $i \notin \Pi_t$  such that

$$\left( \frac{u_i^T (A_t - \hat{\lambda}I)^{-2} u_i}{\mathbf{tr}(A_t - \hat{\lambda}I)^{-1} - \mathbf{tr}(A_t - \lambda I)^{-1}} - \left(u_i^T (A_t - \hat{\lambda}I)^{-1} u_i\right) \right) > 1.$$

This last relation gives

$$\begin{aligned} \mathbf{tr}(A_t - \lambda I)^{-1} &> \mathbf{tr}(A_t - \hat{\lambda}I)^{-1} - \frac{u_i^T (A_t - \hat{\lambda}I)^{-2} u_i}{1 + u_i^T (A_t - \hat{\lambda}I)^{-1} u_i} \\ &= \mathbf{tr}(A_t - \hat{\lambda}I)^{-1} - \mathbf{tr} \left( \frac{(A_t - \hat{\lambda}I)^{-1} u_i u_i^T (A_t - \hat{\lambda}I)^{-1}}{1 + u_i^T (A_t - \hat{\lambda}I)^{-1} u_i} \right) \\ &= \mathbf{tr} \left( A_t - \hat{\lambda}I + u_i u_i^T \right)^{-1}, \end{aligned}$$

where the last line was accomplished with the trace property previously indicated and the Sherman-Morrison formula.  $\square$

**3.2. Lower Bound on  $\lambda_{\min}(A_\ell)$ .** Lemma 3.2 ensures that the UCS algorithm can indeed find all  $\ell$  indices. We now estimate an eigenvalue lower bound on  $A_\ell$ . Let  $\lambda^{(t)}$ ,  $\hat{\lambda}^{(t)}$  and  $\lambda_j^{(t)}$  represent the values of  $\lambda$ ,  $\hat{\lambda}$  and  $\lambda_j$ , respectively, determined in iteration  $t$ . Then note that by the definitions of  $\lambda$  and  $\hat{\lambda}$  we have

$$\lambda^{(0)} < \hat{\lambda}^{(0)} \leq \lambda^{(1)} < \hat{\lambda}^{(1)} \leq \dots \leq \lambda^{(\ell-1)} < \hat{\lambda}^{(\ell-1)}.$$

Define the following quantity and functions:

$$(3.4) \quad \hat{T} \stackrel{def}{=} T \left(1 - \hat{\lambda}^{(\ell-1)}\right),$$

$$g(t) \stackrel{def}{=} \frac{\ell \left(1 - \frac{n}{\hat{T}}\right)}{m - t + \hat{T} - n}, \quad \text{and} \quad F(\hat{T}) \stackrel{def}{=} \frac{\ell \left(1 - \frac{n}{\hat{T}}\right)}{m - \frac{\ell-1}{2} + \hat{T} - n} - \frac{n}{\hat{T}}.$$

To bound  $\lambda_{\min}(A_\ell)$ , we first establish a recurrence relation on  $\hat{\lambda}^{(\ell-1)}$ .

LEMMA 3.3. *After the last iteration of the UCS algorithm, we have*

$$\hat{\lambda}^{(\ell-1)} \geq \left(1 - \hat{\lambda}^{(\ell-1)}\right) \left[ \frac{1}{\ell} \sum_{t=0}^{\ell-1} g(t) - \frac{n}{\hat{T}} \right].$$



*Proof.* Remember that  $T = \mathbf{tr} \left( A - \lambda^{(t)} I \right)^{-1} = \sum_{j=1}^n \frac{1}{\lambda_j^{(t)} - \lambda^{(t)}}$ , and note that

$$(3.5) \quad \frac{1 - \lambda_j^{(t)}}{\lambda_j^{(t)} - \lambda^{(t)}} = \frac{1 - \lambda^{(t)}}{\lambda_j^{(t)} - \lambda^{(t)}} + \frac{\lambda^{(t)} - \lambda_j^{(t)}}{\lambda_j^{(t)} - \lambda^{(t)}} = \frac{1 - \lambda^{(t)}}{\lambda_j^{(t)} - \lambda^{(t)}} - 1.$$

The equation  $f \left( \widehat{\lambda}^{(t)} \right) = 0$  gives

$$\left( \widehat{\lambda}^{(t)} - \lambda^{(t)} \right) \left( m - t + \sum_{j=1}^n \frac{1 - \lambda_j^{(t)}}{\lambda_j^{(t)} - \lambda^{(t)}} \right) = \frac{\sum_{j=1}^n \frac{1 - \lambda_j^{(t)}}{(\lambda_j^{(t)} - \lambda^{(t)}) (\lambda_j^{(t)} - \widehat{\lambda}^{(t)})}}{\sum_{j=1}^n \frac{1}{(\lambda_j^{(t)} - \lambda^{(t)}) (\lambda_j^{(t)} - \widehat{\lambda}^{(t)})}}.$$

Applying equation (3.5) to both sides:

$$\begin{aligned} & \left( \widehat{\lambda}^{(t)} - \lambda^{(t)} \right) \left( m - t + \left( 1 - \lambda^{(t)} \right) T - n \right) \\ &= 1 - \lambda^{(t)} - \frac{\sum_{j=1}^n \frac{1}{\lambda_j^{(t)} - \widehat{\lambda}^{(t)}}}{\sum_{j=1}^n \frac{1}{(\lambda_j^{(t)} - \lambda^{(t)}) (\lambda_j^{(t)} - \widehat{\lambda}^{(t)})}} \\ &\geq 1 - \lambda^{(t)} - \frac{n \left( \max_{j^*} \frac{1}{\lambda_{j^*}^{(t)} - \lambda^{(t)}} \right)}{\left( \max_{j^*} \frac{1}{\lambda_{j^*}^{(t)} - \lambda^{(t)}} \right) \left( \sum_{j=1}^n \frac{1}{\lambda_j^{(t)} - \lambda^{(t)}} \right)} \\ &= 1 - \lambda^{(t)} - \frac{n}{T}. \end{aligned}$$

Since

$$\left( \widehat{\lambda}^{(t-1)} - \lambda^{(t)} \right) \leq 0, \quad \text{and} \quad \left( \widehat{\lambda}^{(t)} - \lambda^{(t)} \right) \geq \frac{1 - \lambda^{(t)} - \frac{n}{T}}{m - t + \left( 1 - \lambda^{(t)} \right) T - n},$$

we have

$$\begin{aligned} \widehat{\lambda}^{(\ell-1)} &\geq \widehat{\lambda}^{(\ell-1)} + \sum_{t=1}^{\ell-1} \overbrace{\left( \widehat{\lambda}^{(t-1)} - \lambda^{(t)} \right)}^{\leq 0} - \lambda^{(0)} + \lambda^{(0)} \\ &= \sum_{t=0}^{\ell-1} \left( \widehat{\lambda}^{(t)} - \lambda^{(t)} \right) + \lambda^{(0)} \\ &\geq \sum_{t=0}^{\ell-1} \frac{1 - \lambda^{(t)} - \frac{n}{T}}{m - t + \left( 1 - \lambda^{(t)} \right) T - n} - \frac{n}{T} \\ (3.6) \quad &\geq \sum_{t=0}^{\ell-1} \frac{1 - \widehat{\lambda}^{(\ell-1)} - \frac{n}{T}}{m - t + \left( 1 - \widehat{\lambda}^{(\ell-1)} \right) T - n} - \frac{n}{T}. \end{aligned}$$

Inequality (3.6) follows by noting that the terms in the sum are decreasing in  $\lambda^{(t)}$ . The final substitution is necessary because solving the preceding recurrence relation is impractical. To further simplify calculations, we define

$$\widehat{T} := T \left( 1 - \widehat{\lambda}^{(\ell-1)} \right).$$

Therefore,

$$\begin{aligned}\widehat{\lambda}^{(\ell-1)} &\geq \left(1 - \widehat{\lambda}^{(\ell-1)}\right) \left[ \sum_{t=0}^{\ell-1} \frac{1 - \frac{n}{\widehat{T}}}{m - t + \widehat{T} - n} - \frac{n}{\widehat{T}} \right] \\ &= \left(1 - \widehat{\lambda}^{(\ell-1)}\right) \left[ \frac{1}{\ell} \sum_{t=0}^{\ell-1} g(t) - \frac{n}{\widehat{T}} \right]. \quad \square\end{aligned}$$

Next, to demonstrate the effectiveness of the algorithm, we derive a lower bound for  $\lambda_n$  after  $\ell$  iterations. This analysis will involve selecting an appropriate  $T$  to maximize the lower bound.

LEMMA 3.4. *If  $\widehat{T} > n$ , then*

$$\lambda_{\min}(A_\ell) \geq \frac{F(\widehat{T})}{1 + F(\widehat{T})}.$$

*Proof.* A key observation is that  $g(t)$  is *strictly* convex in  $t$ , which is easily verified by showing that the second derivative  $\frac{d^2 g}{dt^2}(t)$  is positive by our assumptions that  $\widehat{T} > n$  and  $m \geq \ell > t$ . Next, we apply Jensen's Inequality for discrete sums [21] to the recurrence relation in Lemma 3.3:

$$\begin{aligned}\widehat{\lambda}^{(\ell-1)} &\geq \left(1 - \widehat{\lambda}^{(\ell-1)}\right) \left[ \frac{1}{\ell} \sum_{t=0}^{\ell-1} \frac{\ell \left(1 - \frac{n}{\widehat{T}}\right)}{m - t + \widehat{T} - n} - \frac{n}{\widehat{T}} \right] \\ &> \left(1 - \widehat{\lambda}^{(\ell-1)}\right) \left[ \frac{\ell \left(1 - \frac{n}{\widehat{T}}\right)}{\frac{1}{\ell} \sum_{t=0}^{\ell-1} m - t + \widehat{T} - n} - \frac{n}{\widehat{T}} \right] \\ &= \left(1 - \widehat{\lambda}^{(\ell-1)}\right) \left[ \frac{\ell \left(1 - \frac{n}{\widehat{T}}\right)}{m - \frac{\ell-1}{2} + \widehat{T} - n} - \frac{n}{\widehat{T}} \right] \\ &= \left(1 - \widehat{\lambda}^{(\ell-1)}\right) F(\widehat{T}).\end{aligned}$$

Along with  $\lambda_n > \widehat{\lambda}$  from Lemma 3.1, this finally leads to

$$(3.7) \quad \lambda_{\min} = \lambda_n > \widehat{\lambda}^{(\ell-1)} > \frac{F(\widehat{T})}{1 + F(\widehat{T})}. \quad \square$$

The expression on the right-hand side of (3.7) is monotonically increasing in  $F$ . So, maximizing  $F(\widehat{T})$  will also maximize the lower bound on  $\widehat{\lambda}^{(\ell-1)}$ .

LEMMA 3.5. *The function  $F(\widehat{T})$  is maximized at*

$$\widehat{T}^* = \frac{n \left(m + \frac{\ell+1}{2} - n\right) + \sqrt{n\ell \left(m - \frac{\ell-1}{2}\right) \left(m + \frac{\ell+1}{2} - n\right)}}{\ell - n}.$$

*Proof.* Setting the derivative of  $F(\hat{T})$  to zero:

$$\frac{dF}{d\hat{T}} = \frac{(n-\ell)T^2 + 2n\left(m + \frac{\ell+1}{2} - n\right)T + n\left(m + \frac{\ell+1}{2} - n\right)\left(m - \frac{\ell-1}{2} - n\right)}{\hat{T}^2\left(m - \frac{\ell-1}{2} - n + \hat{T}\right)^2} = 0.$$

Solving for the desired root:

$$\hat{T}^* = \frac{n\left(m + \frac{\ell+1}{2} - n\right) + \sqrt{n\ell\left(m - \frac{\ell-1}{2}\right)\left(m + \frac{\ell+1}{2} - n\right)}}{\ell - n}.$$

We see that  $\hat{T}^*$  is the global maximum on the region  $\hat{T} \in (n, \infty)$  via the first derivative test since  $\frac{dF}{d\hat{T}} > 0$  for  $n < \hat{T} < \hat{T}^*$  and  $\frac{dF}{d\hat{T}} < 0$  for  $\hat{T}^* < \hat{T}$ .  $\square$

We remark that combining (3.4) and (3.7) implies that the UCS algorithm should choose  $T = \hat{T}^*\left(1 + F(\hat{T}^*)\right)$  for effective column selection. We are now ready to estimate  $\lambda_{\min}(A_\ell)$ .

**THEOREM 3.6.** *If  $T$  is chosen according to Lemma 3.5 in the UCS algorithm, then*

$$\lambda_{\min}(A_\ell) > \frac{1}{\kappa},$$

where  $\kappa$  is defined in (2.4).

*Proof.* We wish to apply our choice of  $\hat{T}$  to Lemma 3.4. We satisfy the assumption

$$\hat{T}^* = \frac{n\left(m + \frac{\ell+1}{2} - n\right) + \sqrt{n\ell\left(m - \frac{\ell-1}{2}\right)\left(m + \frac{\ell+1}{2} - n\right)}}{\ell - n} \geq \frac{n(m-n)}{\ell - n} \geq n.$$

Therefore, plugging  $\hat{T}^*$  into (3.7) of Lemma 3.4:

$$\begin{aligned} \lambda_{\min}(A_\ell) &> \frac{F(\hat{T})}{1 + F(\hat{T})} \\ &= \frac{(\ell - n)\hat{T} - n\left(m + \frac{\ell+1}{2} - n\right)}{\hat{T}\left(m - \frac{\ell-1}{2} - n + \hat{T}\right) + (\ell - n)\hat{T} - n\left(m + \frac{\ell+1}{2} - n\right)} \\ &= \frac{(\ell - n)^2}{\left(\sqrt{n\left(m + \frac{\ell+1}{2} - n\right)} + \sqrt{\ell\left(m - \frac{\ell+1}{2}\right)}\right)^2 + (\ell - n)^2}. \quad \square \end{aligned}$$

**3.3. Correctness of the Unweighted Column Selection Algorithm.** We are now in a position to prove Theorem 2.1. Our arguments are similar to those of the weighted sparsifier algorithm in [3].

*Proof of Theorem 2.1.* By Proposition 1.3, we only need to show  $\frac{1}{\kappa}L_G \preceq L_H$ . Consider the SVD of  $W_G^{1/2}B_G$  in equation (1.1), and let  $x$  be any vector such that  $y = \Sigma_G V_G x \neq 0$ . Then

$$\begin{aligned} L_G &= B_G^T W_G B_G = V_G^T \Sigma_G^2 V_G, \\ L_H &= B_G^T W_H B_G = B_G^T W_G^{1/2} \Pi_\ell^T \Pi_\ell W_G^{1/2} B_G \\ &= V_G^T \Sigma_G (U_G \Pi_\ell^T \Pi_\ell U_G^T) \Sigma_G V_G. \end{aligned}$$

It follows that

$$(3.8) \quad \begin{aligned} \frac{x^T L_H x}{x^T L_G x} &= \frac{x^T (V_G^T \Sigma_G (U_G \Pi_\ell^T \Pi_\ell U_G^T) \Sigma_G V_G) x}{x^T (V_G^T \Sigma_G^2 V_G) x} \\ &= \frac{y^T U_G \Pi_\ell^T \Pi_\ell U_G^T y}{y^T y}. \end{aligned}$$

On the other hand, by construction we have

$$A_\ell = \sum_{j \in \Pi_\ell} u_j u_j^T = U_G \Pi_\ell^T \Pi_\ell U_G^T.$$

With equation (3.8), the Courant-Fisher min-max property gives

$$\frac{x^T L_H x}{x^T L_G x} = \frac{y^T U_G \Pi_\ell^T \Pi_\ell U_G^T y}{y^T y} \geq \lambda_{\min}(A_\ell) > \frac{1}{\kappa},$$

where the last line is due to Theorem 3.6.  $\square$

**4. Performance Comparison of UCS and Other Algorithms.** This section compares the bound (2.4) to bounds of other current methods.

**4.1. Comparison with Twice-Ramanujan Sparsifiers.** Given a weighted graph  $G = (V, E, w)$ , the algorithm of [3] produces a sparsified graph  $H = (V, F, \hat{w})$ , where  $F$  is a subset of  $E$  and  $\hat{w}$  contains new edge weights, such that

$$(4.1) \quad L_G \preceq L_H \preceq \left( \frac{\sqrt{d} + 1}{\sqrt{d} - 1} \right)^2 L_G,$$

where the parameter  $d$  is defined via the equation  $\ell = \lceil d(n-1) \rceil$ .

By choosing  $d$  to be a moderate and dimension-independent constant, equation (4.1) asserts that every graph  $G = (V, E, w)$  has a *weighted* spectral sparsifier with a number of edges linear in  $|V|$ . This strong result, nevertheless, is obtained by allowing unrestricted changes in the graph weights. Such changes may be undesirable, especially if  $G$  is unweighted, and the UCS algorithm may be preferred.

To compare the effectiveness of these two types of sparsifiers, we simplify equation (2.4):

$$\frac{1}{\kappa} \approx \frac{(\sqrt{d} - 1)^2}{m/n + d/2 + (\sqrt{d} - 1)^2}.$$

It follows that for  $\kappa = \Theta(1)$ , a dimension-independent constant, we must choose  $d = \Theta(m/n)$ . This is the price one must pay to retain the original weights. For  $d \ll m/n$ , the UCS algorithm computes a sparsified graph with a  $\kappa$  that grows at most linearly with  $m/n$ . The algorithm of [3] runs in time  $O(dn^3m)$ , which is equivalent to UCS.

**4.2. Further Comparisons of Column Selection Algorithms.** The algorithm of [3] has been generalized in [6] to a column selection algorithm for computing CX decompositions. In this work, Boutsidis, Drineas, and Magdon-Ismail prove that, given row-orthonormal matrices  $V_1^T = (\bar{v}_1^1 \ \bar{v}_2^1 \ \dots \ \bar{v}_m^1) \in \mathbb{R}^{n \times m}$  and

$V_2^T = (\bar{v}_1^2 \ \bar{v}_2^2 \ \cdots \ \bar{v}_m^2) \in \mathbb{R}^{(m-n) \times m}$  then for a given  $n < \ell \leq m$  there exist weights  $s_i \geq 0$  with at most  $\ell$  of them nonzero such that

$$(4.2) \quad \lambda_n \left( \sum_{i=1}^m s_i \bar{v}_i^1 (\bar{v}_i^1)^T \right) \geq \left( 1 - \sqrt{\frac{n}{\ell}} \right)^2$$

and

$$(4.3) \quad \lambda_1 \left( \sum_{i=1}^m s_i \bar{v}_i^2 (\bar{v}_i^2)^T \right) \leq \left( 1 + \sqrt{\frac{m-n}{\ell}} \right)^2.$$

In the context of CX decompositions,  $\begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = V^T \in \mathbb{R}^{m \times m}$  is understood to be the loadings matrix of a data matrix  $A$ , i.e.  $A = U \Sigma V^T$  is the SVD of  $A$  (although the algorithm could be applied to other matrices for other applications). Their work includes an algorithm for finding the weights, Deterministic Dual Set Spectral Sparsification (DDSSS).

**THEOREM 4.1.** *Let  $\Pi_\ell^{DDSSS} \in \mathbb{R}^{m \times \ell}$  denote a matrix that chooses the  $\ell$  columns selected by the DDSSS algorithm. The inequalities (4.2) and (4.3) imply*

$$\begin{aligned} \sigma_{\min}^2(V_1^T \Pi_\ell^{DDSSS}) &\geq \frac{(\sqrt{\ell} - \sqrt{n})^2}{(\sqrt{\ell} + \sqrt{m-n})^2 + (\sqrt{\ell} - \sqrt{n})^2} \\ &\stackrel{\text{def}}{=} \frac{1}{\kappa_{DDSSS}}. \end{aligned}$$

*Proof.* We interpret these inequalities as a bound on  $\lambda_n$  by first partitioning

$$V^T \Pi = \begin{pmatrix} V_1 & V_1' \\ V_2 & V_2' \end{pmatrix},$$

where  $\Pi$  is a permutation matrix that orders the selected columns first. Then, using a CS decomposition [13], we can write

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} P_1 (C \ 0) Q_1^T \\ P_2 \begin{pmatrix} -S & 0 \\ 0 & I \\ 0 & 0 \end{pmatrix} Q_2^T \end{pmatrix},$$

where  $C$  and  $S$  are diagonal matrices with non-negative entries such that  $C^2 + S^2 = I$ . Furthermore, because  $P_1$  and  $Q$  are orthogonal, by inspection  $C$  contains the singular values of  $V_1$ . Hence

$$\lambda_n \geq \sigma_{\min}^2(V_1) = \sigma_{\min}^2(C).$$

Now let  $W$  be a weight matrix, whose diagonal entries are  $\sqrt{s_i}$ , the weights from above. Define

$$\begin{aligned} \hat{Q} &\stackrel{\text{def}}{=} Q_1^T (W W^T) Q_1 \\ &\stackrel{\text{def}}{=} \begin{pmatrix} \hat{Q}_{11} & \hat{Q}_{12} \\ \hat{Q}_{21} & \hat{Q}_{22} \end{pmatrix}. \end{aligned}$$

Then

$$\begin{aligned}
& (V_2 W) (V_1 W)^\dagger \\
&= V_2 W (V_1 W)^T (V_1 W W^T V_1^T)^{-1} \\
&= V_2 (W W^T) V_1^T (V_1 (W W^T) V_1^T)^{-1} \\
&= V_2 (W W^T) (P_1 \begin{pmatrix} C & 0 \end{pmatrix} Q_1^T)^T (P_1 \begin{pmatrix} C & 0 \end{pmatrix} Q_1^T (W W^T) Q_1 \begin{pmatrix} C \\ 0 \end{pmatrix} P_1^T)^{-1} \\
&= P_2 \begin{pmatrix} -S & 0 \\ 0 & I \\ 0 & 0 \end{pmatrix} \hat{Q} \begin{pmatrix} C \\ 0 \end{pmatrix} P_1^T P_1 \left( \begin{pmatrix} C & 0 \end{pmatrix} \hat{Q} \begin{pmatrix} C \\ 0 \end{pmatrix} \right)^{-1} P_1^T \\
&= P_2 \begin{pmatrix} -S & 0 \\ 0 & I \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{Q}_{11} & \hat{Q}_{12} \\ \hat{Q}_{21} & \hat{Q}_{22} \end{pmatrix} \begin{pmatrix} C \\ 0 \end{pmatrix} (C \hat{Q}_{11} C)^{-1} P_1^T \\
&= P_2 \begin{pmatrix} -S & 0 \\ 0 & I \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{Q}_{11} C \\ \hat{Q}_{21} C \end{pmatrix} C^{-1} \hat{Q}_{11}^{-1} C^{-1} P_1^T \\
&= P_2 \begin{pmatrix} -S C^{-1} \\ \hat{Q}_{21} \hat{Q}_{11}^{-1} C^{-1} \\ 0 \end{pmatrix} P_1^T.
\end{aligned}$$

Therefore

$$\begin{aligned}
\sqrt{\frac{1 - \sigma_{\min}^2(C)}{\sigma_{\min}^2(C)}} &= \|S C^{-1}\|_2 \\
&\leq \|(V_2 W) (V_1 W)^\dagger\|_2 \\
&\leq \left(1 + \sqrt{\frac{m-n}{\ell}}\right) \left(1 - \sqrt{\frac{n}{\ell}}\right)^{-1}.
\end{aligned}$$

Rearranging

$$\begin{aligned}
\sigma_{\min}^2(C) &\geq \frac{(1 - \sqrt{\frac{n}{\ell}})^2}{\left(1 + \sqrt{\frac{m-n}{\ell}}\right)^2 + (1 - \sqrt{\frac{n}{\ell}})^2} \\
&= \frac{(\sqrt{\ell} - \sqrt{n})^2}{(\sqrt{\ell} + \sqrt{m-n})^2 + (\sqrt{\ell} - \sqrt{n})^2}. \quad \square
\end{aligned}$$

COROLLARY 4.2. *Let  $\kappa_{UCS}$  be as defined in equation (2.4). Then*

$$\frac{1}{\kappa_{UCS}} > \frac{1}{\kappa_{DDSSS}}.$$

*Proof.*

$$\begin{aligned}
& \frac{(\ell - n)^2}{\left( \sqrt{n \left( m + \frac{\ell+1}{2} - n \right)} + \sqrt{\ell \left( m - \frac{\ell+1}{2} \right)} \right)^2 + (\ell - n)^2} \\
&= \frac{(\sqrt{\ell} - \sqrt{n})^2}{\left( \frac{\sqrt{n \left( m + \frac{\ell+1}{2} - n \right)} + \sqrt{\ell \left( m - \frac{\ell+1}{2} \right)}}{\sqrt{\ell} + \sqrt{n}} \right)^2 + (\sqrt{\ell} - \sqrt{n})^2} \\
&\geq \frac{(\sqrt{\ell} - \sqrt{n})^2}{\left( \frac{\sqrt{n \left( m + \frac{\ell+1}{2} - n \right)} + \sqrt{\ell m}}{\sqrt{\ell} + \sqrt{n}} \right)^2 + (\sqrt{\ell} - \sqrt{n})^2} \\
&\geq \frac{(\sqrt{\ell} - \sqrt{n})^2}{\left( \frac{\sqrt{n(m+\ell-n)} + \sqrt{\ell m}}{\sqrt{\ell} + \sqrt{n}} \right)^2 + (\sqrt{\ell} - \sqrt{n})^2} \\
&\geq \frac{(\sqrt{\ell} - \sqrt{n})^2}{\left( \frac{\sqrt{n(m+\ell-n)} + \sqrt{\ell m - n\ell + \ell^2}}{\sqrt{\ell} + \sqrt{n}} \right)^2 + (\sqrt{\ell} - \sqrt{n})^2} \\
&\geq \frac{(\sqrt{\ell} - \sqrt{n})^2}{\left( \frac{\sqrt{nm - n^2} + \sqrt{n\ell} + \sqrt{\ell m - n\ell + \ell^2}}{\sqrt{\ell} + \sqrt{n}} \right)^2 + (\sqrt{\ell} - \sqrt{n})^2} \\
&= \frac{(\sqrt{\ell} - \sqrt{n})^2}{\left( \frac{(\sqrt{m-n} + \sqrt{\ell})(\sqrt{\ell} + \sqrt{n})}{\sqrt{\ell} + \sqrt{n}} \right)^2 + (\sqrt{\ell} - \sqrt{n})^2} \\
&= \frac{(\sqrt{\ell} - \sqrt{n})^2}{(\sqrt{m-n} + \sqrt{\ell})^2 + (\sqrt{\ell} - \sqrt{n})^2}. \quad \square
\end{aligned}$$

This suggests the UCS algorithm may find a better subset than the column selection algorithm in [6]. Observe that typically  $m \gg \ell \geq n$ . For the purpose of finding a well-conditioned subset of columns in  $V_1^T \in \mathbb{R}^{n \times m}$ , requiring the whole matrix  $V^T \in \mathbb{R}^{m \times m}$  is computationally expensive. On the other hand, an even better subset can be obtained by applying the UCS algorithm directly to  $V_1^T$ , at considerable savings in computational time and memory usage. This algorithm runs in time  $O\left(\ell m \left(n^2 + (m - \ell)^2\right)\right) \approx O(\ell m^3)$ , far slower than UCS.

**5. A Numeric Example: Graph Visualization.** We test the UCS algorithm on the Autonomous systems AS-733 dataset in [12]<sup>2</sup>. The data is undirected, unweighted, and contains 493 nodes and 1189 edges. To visualize the data, nodes are plotted using coordinates determined by the force-directed Fruchterman-Reingold algorithm. This algorithm treats the edges of a graph as forces (similar to springs), and perturbs node coordinates until the graph appears to be near an equilibrium state [9].

We apply the force-directed algorithm with two methodologies. First, the force-directed algorithm is run on the whole graph to determine a fixed set of node coordinates. Using these coordinates, the original graph is plotted with various sparsifiers in Figure 2. Second, we run the force-directed algorithm on each sparsifier to determine node coordinates for that sparsifier, and plot both the sparsifier and the original graph on these coordinates (Figure 3). While this requires rerunning the force-directed al-

<sup>2</sup>File as19981229



Fig. 1: Autonomous System Example: Original Graph

gorithm for each sparsifier, the algorithm converges faster because of the reduced number of edges.

Although the original graph can be considered sparse, visualization of the graph is difficult. In Figure 1, a few nodes are seen to have high degree, but little information is readily available about important edges in the graph or about how important nodes are related. Figure 2 shows that plotting the sparsifier on the original graph provides incremental benefit. The sparser graphs begin to highlight important nodes and important edges connecting them, but visualization remains difficult. Rerunning the force-directed algorithm on the sparsifiers, nevertheless, evokes an easily interpretable structure, where important nodes, clusters, and important edges connecting clusters are readily visible (Figure 3).

**6. Relationship to the Kadison-Singer Problem.** Let  $p \geq 2$  be an integer, and let  $U = (u_1, \dots, u_m) \in \mathbb{R}^{n \times m}$  be a matrix that satisfies

$$(6.1) \quad \sum_{k=1}^n u_k u_k^T = I, \quad \text{and} \quad \|u_k\|_2 \leq \delta, \quad \text{for } k = 1, \dots, m,$$

where  $0 < \delta < 1$ . Equation (6.1) implies that  $U$  is a row-orthonormal matrix and that each column of  $U$  is uniformly bounded away from 1 in 2-norm. Marcus *et al.* [14] show that there exists a partition

$$(6.2) \quad \mathcal{P} = \mathcal{P}_1 \cup \dots \cup \mathcal{P}_p$$

of  $\{1, \dots, n\}$  such that

$$\|U(:, \mathcal{P}_k)\|_2 \leq \frac{1}{\sqrt{p}} + \delta, \quad \text{for } k = 1, \dots, p.$$

When the graph  $G$  is sufficiently dense, equation (6.2) implies the existence of an unweighted graph sparsifier (see Batson, *et al.* [3]).



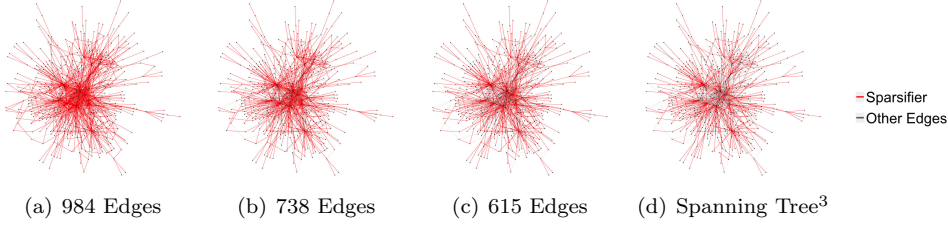


Fig. 2: Autonomous Systems Graph with Sparsifiers of Various Cardinalities (node coordinates calculated from whole graph)

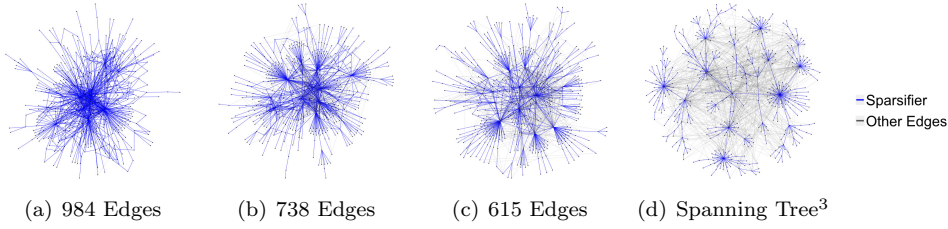


Fig. 3: Autonomous Systems Graph with Sparsifiers of Various Cardinalities (node coordinates recalculated for each sparsifier)

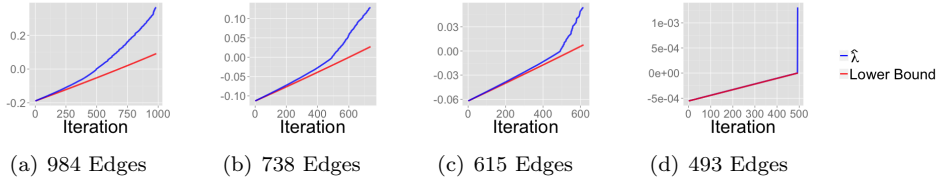


Fig. 4: Progress During Iteration and Theoretical Singular Value Lower Bound for Sparsifiers of Various Cardinalities

**7. Conclusion.** We have presented an efficient algorithm for the construction of unweighted spectral sparsifiers for general weighted and unweighted graphs, addressing the open question of the existence of such graph sparsifiers for general graphs [3]. Our algorithm is supported by strong theoretical spectral bounds. Through numeric experiments, we have demonstrated that our sparsification algorithm can be an effective tool for graph visualization, and anticipate that it will prove useful for wide-ranging applications involving large graphs. An important feature of our sparsification algorithm is the deterministic unweighted column selection algorithm on which it is based. An open question is the existence of a larger lower spectral bound, either with the same  $T$  or a new one.

<sup>3</sup>Calculated by running UCS algorithm with  $\ell = 493$  and omitting the final edge. In general, a spanning tree for a connected graph can be found by selecting  $\ell = n + 1$  edges and removing an edge from a loop created by the UCS algorithm.

## REFERENCES

- [1] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, pages 5–14. ACM, 2012.
- [2] H. Avron and C. Boutsidis. Faster subset selection for matrices and applications. *CoRR*, abs/1201.0127, 2012.
- [3] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012.
- [4] J. D. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng. Spectral sparsification of graphs: theory and algorithms. *Commun. ACM*, 56(8):87–94, 2013.
- [5] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in  $O(n^2)$  time. In Gary L. Miller, editor, *STOC*, pages 47–55. ACM, 1996.
- [6] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. *CoRR*, abs/1103.0995, 2011.
- [7] F. Chierichetti, S. Lattanzi, and A. Panconesi. Rumour spreading and graph conductance. In *SODA*, pages 1657–1663. SIAM, 2010.
- [8] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 273–282. ACM, 2011.
- [9] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software—Practice & Experience*, 21(11):1129–1164, 1991.
- [10] M. Kapralov and R. Panigrahy. Spectral sparsification via random spanners. In Shafi Goldwasser, editor, *ITCS*, pages 393–398. ACM, 2012.
- [11] C. Leiserson. Fat-trees: Universal Networks for Hardware-efficient Supercomputing. *IEEE Trans. Comput.*, 34(10):892–901, 1985.
- [12] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, October 2014.
- [13] C. Van Loan. Computing the cs and the generalized singular value decompositions. *Numerische Mathematik*, 46, Issue 4:479–491, 1985.
- [14] A. W. Marcus, D. A. Spielman, and N. Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *CoRR*, abs/1306.3969, 2014.
- [15] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In Chid Apt, Joydeep Ghosh, and Padhraic Smyth, editors, *KDD*, pages 529–537. ACM, 2011.
- [16] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.
- [17] D. A. Spielman and S.-H. Teng. Solving sparse, symmetric, diagonally-dominant linear systems in time  $O(m^{1.31})$ . *CoRR*, cs.DS/0310036, 2003.
- [18] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC’04*, pages 81–90, 2004.
- [19] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.
- [20] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *CoRR*, abs/0808.4134, 2008.
- [21] V. A. Zorich. *Mathematical Analysis I*. Springer, Berlin, 2004.